MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA

TECHNICAL UNIVERSITY OF MOLDOVA

FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS

DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

PW Lab1. Learn CSS and HTML

Project report

Mentor: prof., Alexei Sersun

Students: Covtun Serghei, FAF-203

Chișinău, 2023

Abstract

Imagine, you're a frontend developer from Melbourne, Australia. You and your company Sleep2nigth Pty Ltd are working on a brand new product - smart To-do lists!

You've pitched the idea in the local startup accelerator and you've won first round of investitions. The only thing what's left - to build the app itself..

Content

Introduction

Your task for this lab is:

1. Copy index.html, style.css and [app.js](app.js) to your repo;
2. Modify them to build an application for to-do list;
3. The app has to cover basic needs:
   - to add to the list;
   - to remove from the list;
   - to mark as done;
   - see "done" and "to-do" lists separately.

4. The app has to look attractive.

Link to Git repo: https://github.com/siorkis/PW/tree/main/lab3

Work

I used HTML, CSS and JS in order to perform provided task.

0.1 Add/remove/done mechanism.

In order to create `add` mechanism, I have used localstore. My “to do” values are just strings which are stored inside localstore. This helps not only easy to access required data, but also save them even after refreshing page.

```
function saveLocalTodos(todo) {
    let todos;
    if(localStorage.getItem("todos") === null) {
        todos = [];
    } else {
        todos = JSON.parse(localStorage.getItem("todos"));
    }
    todos.push(todo);
    localStorage.setItem("todos", JSON.stringify(todos));
}
```

Fig 1. Adding values to localstore.

```
function removeLocalTodos(todo) {
    let todos;
    if(localStorage.getItem("todos") === null) {
        todos = [];
    } else {
        todos = JSON.parse(localStorage.getItem("todos"));
    }

    const todoIndex = todo.children[0].innerText;
    todos.splice(todos.indexOf(todoIndex), 1);
    localStorage.setItem("todos", JSON.stringify(todos));
}
```

Fig 2. Removing values from localstore.

Removing mechanism is very similar to add, but in this case we just removing elements from localstore list.

```
if(item.classList[0] === "complete-btn") {
    const todo = item.parentElement;
    todo.classList.toggle("completed");
}
```

Fig 3. Complete mechanism.

Complete mechanism is nothing else then style solution, which by triggering specific event adds corresponding style.

0.2 Separate lists.

```
switch(e.target.value) {
    case "all":
        todo.style.display = "flex";
        break;
    case "completed":
        if(todo.classList.contains("completed")) {
            todo.style.display = "flex";
        } else {
            todo.style.display = "none";
        }
        break;
    case "incomplete":
        if(!todo.classList.contains("completed")) {
            todo.style.display = "flex";
        } else {
            todo.style.display = "none";
        }
        break;
```

Fig 4. Separate logic.

Lists are separated by class `completed`, so if user wants to see all/completed/incomplete tasks, program just show only those which have required class inside.

0.3 Bonus points tasks.

I have implemented notification mechanism, save/upload mechanism and added possibility to set priority to the task.

```
var audio = new Audio('reminder_sound.mp3');

setTimeout(function() {
    audio.play();
}, 5000);
```

Fig 5. Notification implementation.

Notifications are represented as reminders in this laboratory work, which toggles after required amount of time after loading page and after adding new task to the list.

```
todoList = JSON.parse(localStorage.getItem("todos"));
todoList = todoList.toString();
copy_value = btoa(unescape(encodeURIComponent(todoList)));

navigator.clipboard.writeText(copy_value);
```

Fig 6.1. Copy to do list.

User can copy his to do list in order to save it or to share. Program gets list of values from localstore and transforms it to the encoded string and then saves it to the clipboard.

```
let todo_value = decodeURIComponent(escape(window.atob(code)));
let todo_array = todo_value.split(",");
// console.log(todo_array);
todo_array.forEach(saveLocalTodos);
location.reload();
```

Fig 6.2. Upload to do list.

If user have code of some to do list, he can easily insert it by entering this code into pop-up form. Program decodes that code and inserts values to the localstore.

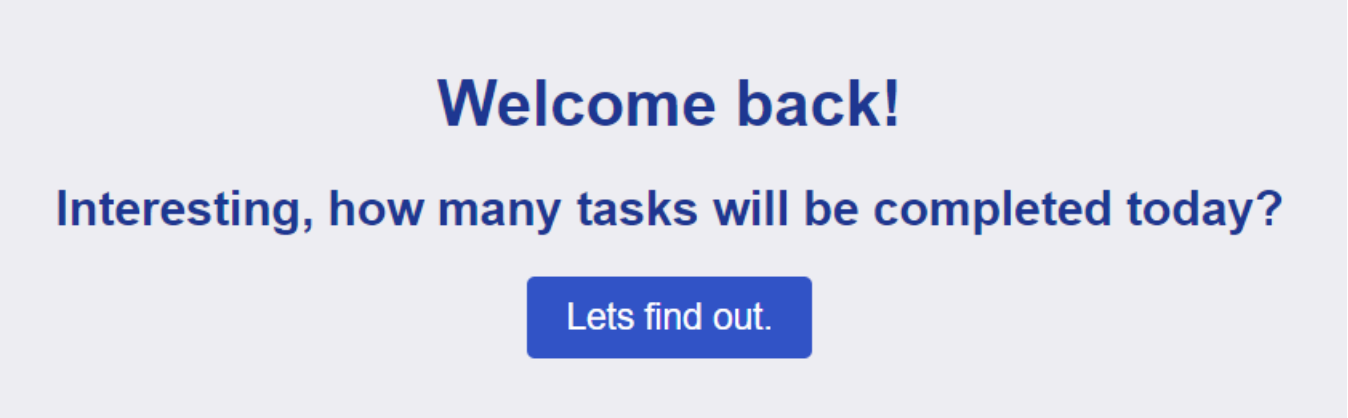
```
todos.sort((a, b) => {
  const countA = (a.match(/!/g) || []).length; // coun
  const countB = (b.match(/!/g) || []).length; // coun
  return countB - countA; // sort in descending order
});
todos.forEach(function(todo) { todoContent(todo); });
```

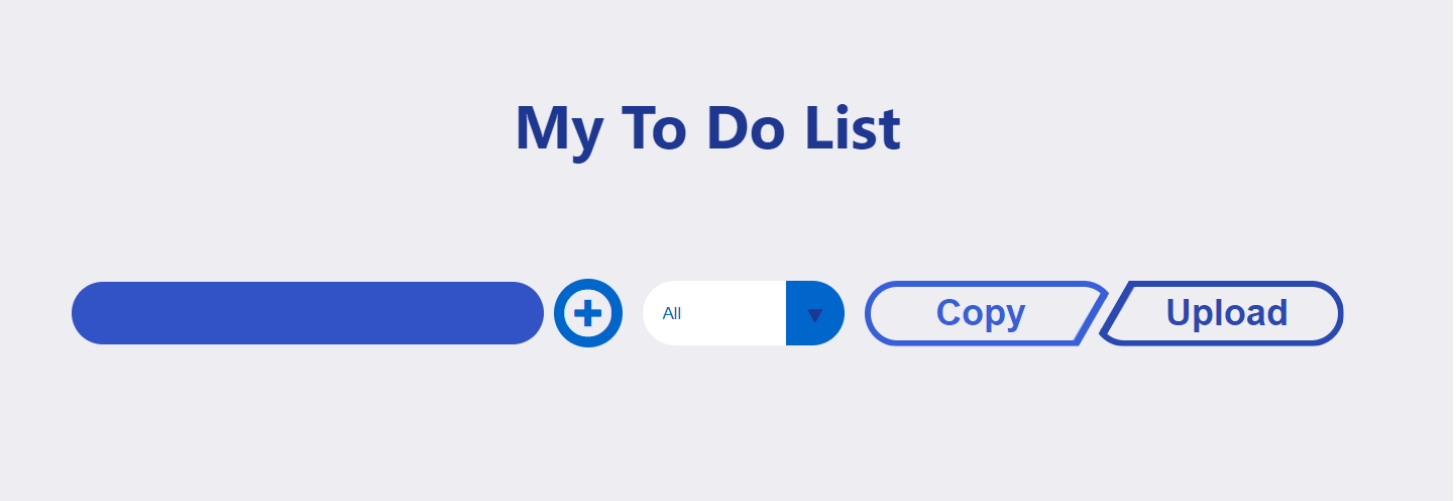
Fig 7. Priority.

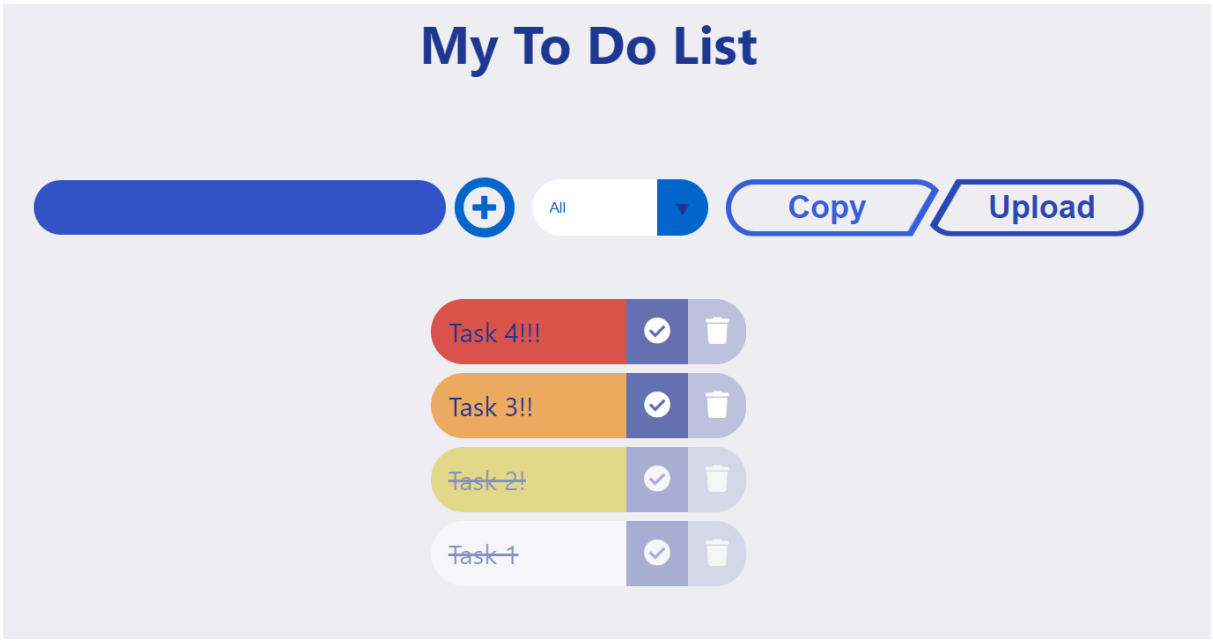
Priority mechanism was done by adding `!` signs to the input string. There are 3 levels of priority. From 1 to 3 of `!` signs. Depending of how much exclamation sign contains inside to do value, this row gets yellow, orange or red color and also priority defines position of this task inside to do list: priority level 3 are shown first.

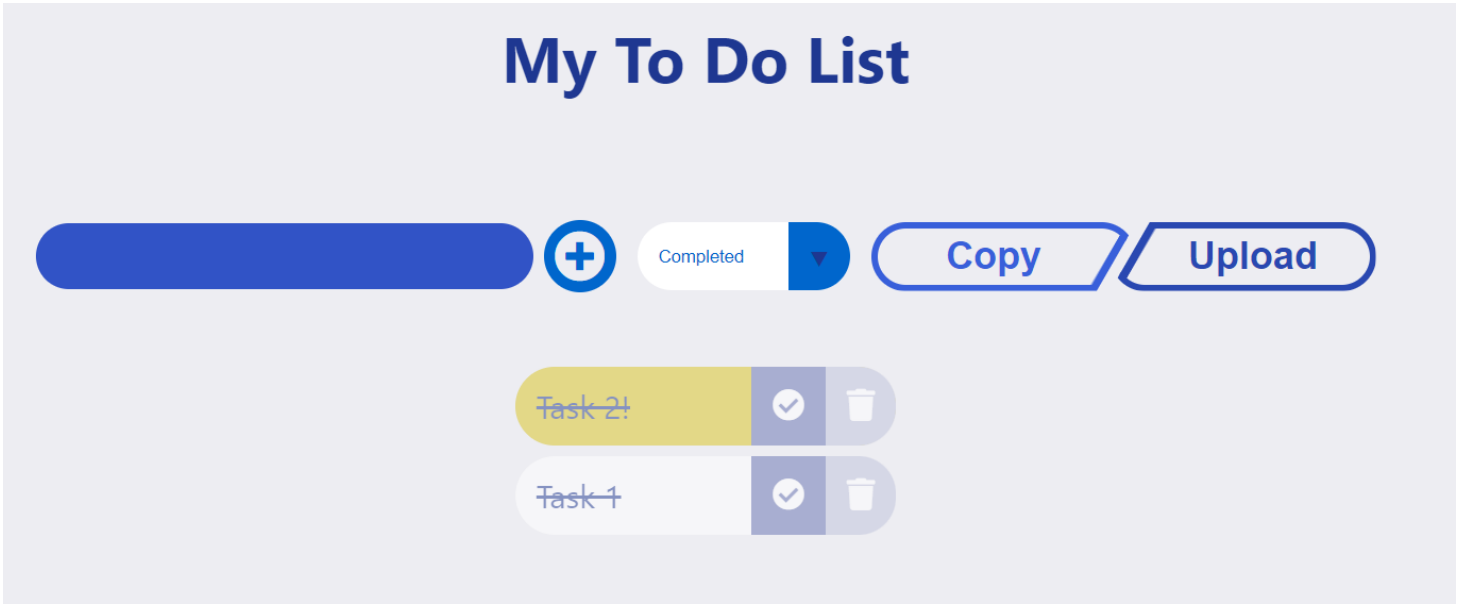
0.4 General overview.

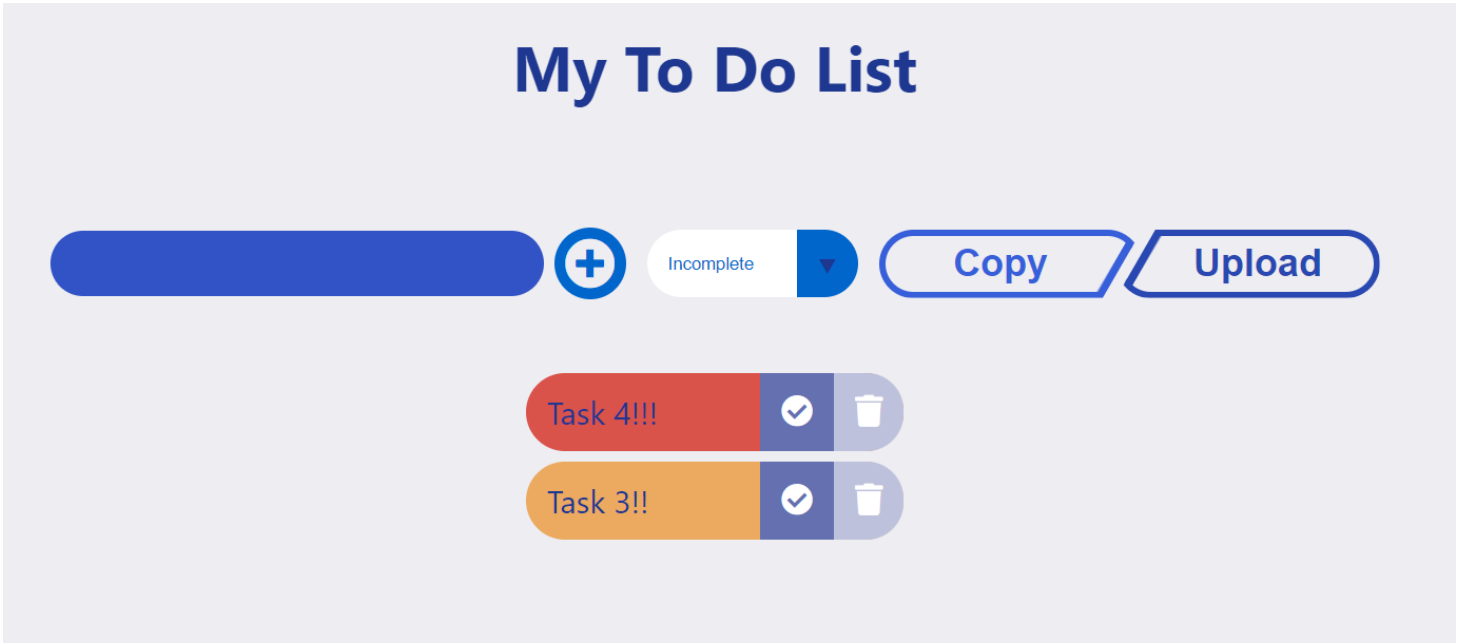
In this chapter are shown main implemented tasks in action.

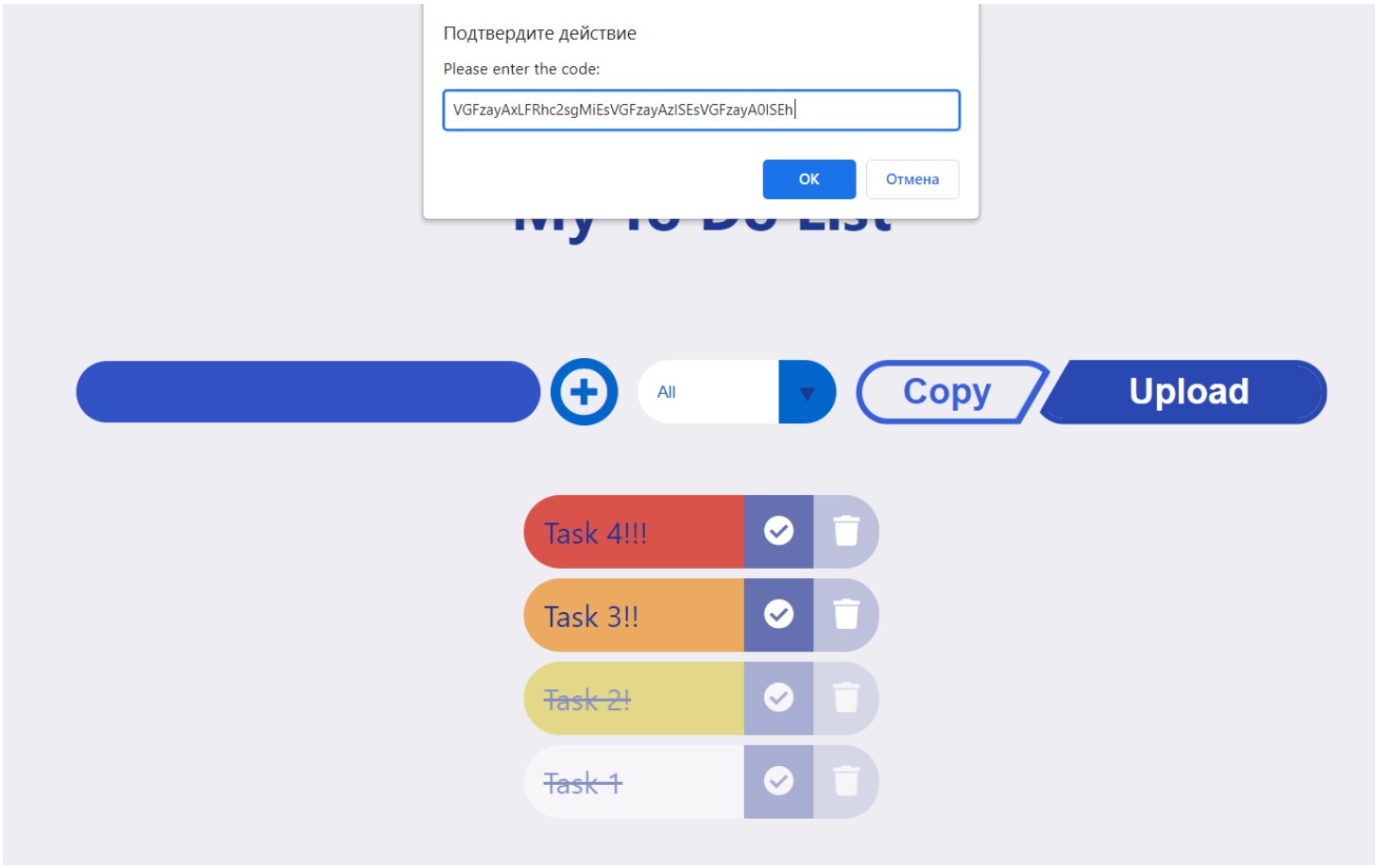


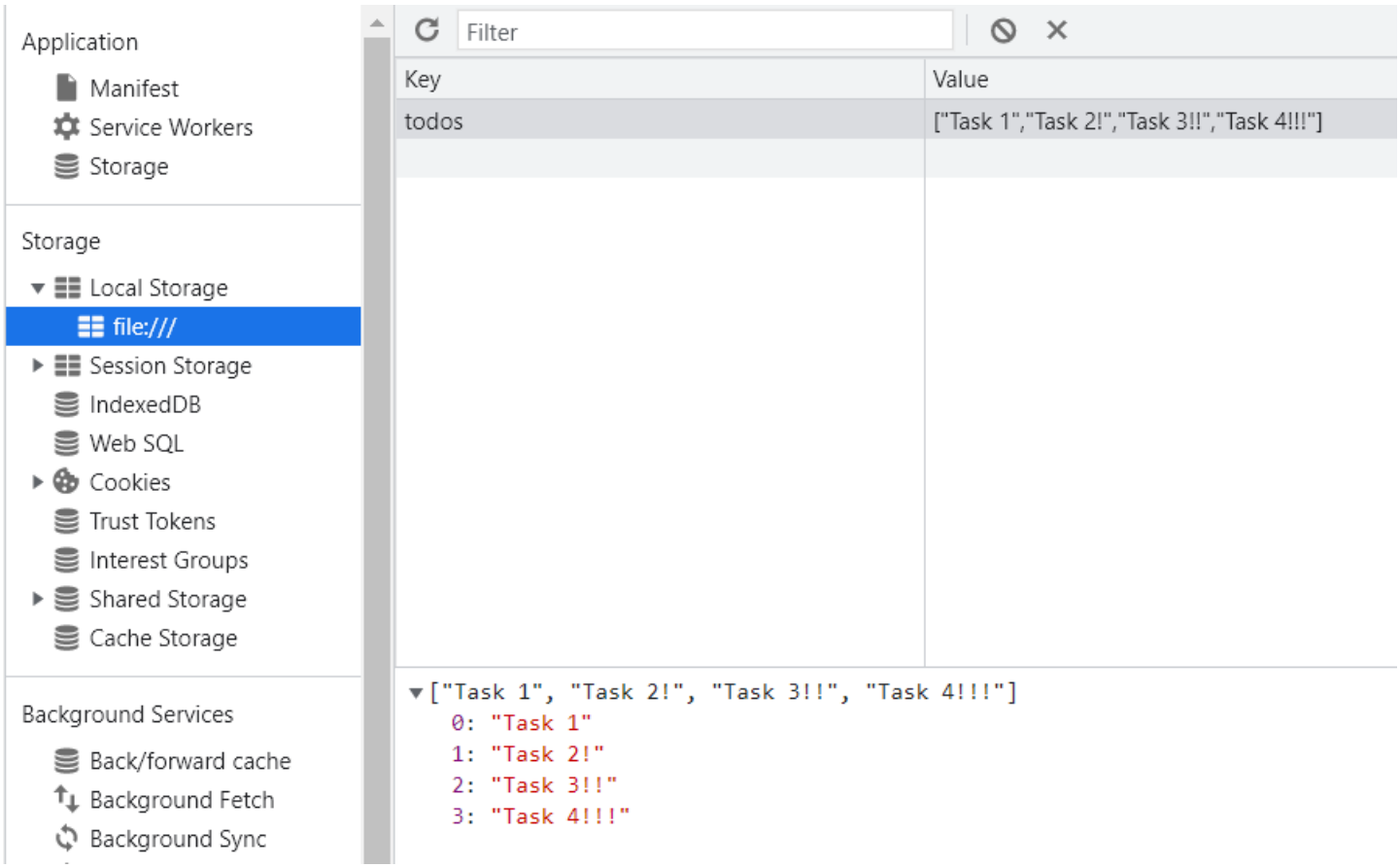



My To Do List
All
Copy
Upload
Task 4!!!
Task 3!!
Task 2!
Task 1


My To Do List
Completed
Copy
Upload
Task 2!
Task 1


My To Do List
Incomplete
Copy
Upload
Task 4!!!
Task 3!!


Подтвердите действие
Please enter the code:
VGFzayAxLFRhc2sgMiEsVGFzayAzISEsVGFzayA0ISEh
OK
Отмена
All
Copy
Upload
Task 4!!!
Task 3!!
Task 2!
Task 1


Application
Manifest
Service Workers
Storage
Storage
Local Storage
file:///
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Interest Groups
Shared Storage
Cache Storage
Background Services
Back/forward cache
Background Fetch
Background Sync
Filter
Key
Value
todos
["Task 1","Task 2!","Task 3!!","Task 4!!!"]
["Task 1", "Task 2!", "Task 3!!", "Task 4!!!"]
0: "Task 1"
1: "Task 2!"
2: "Task 3!!"
3: "Task 4!!!"

Bibliography

[1] W3Schools *Google*, https://www.w3schools.com/css/default.asp

[2] W3Schools *Google*, https://www.w3schools.com/js/default.asp

[3] StackOverflow *Google*, https://stackoverflow.com/questions/16605769/simple-javascript-encryption-decryption-without-using-key

[4] FreeCodecamp *Google*, https://www.freecodecamp.org/news/copy-text-to-clipboard-javascript/

[5] StackOverflow *Google*, https://stackoverflow.com/questions/23223718/failed-to-execute-btoa-on-window-the-string-to-be-encoded-contains-characte

[6] StackOverflow *Google*, https://stackoverflow.com/questions/9419263/how-to-play-audio